Full-Chain Benchmarking for Open Architecture Airborne ISR Systems

A Case Study for GMTI Radar Applications

Matthias Beebe, Matthew Alexander, Paul Foley, Denise Galejs, Stephen Mooney, Iulian Popescu, Kevin Rottman, and Meryl Stav
Embedded and Open Systems Group
MIT Lincoln Laboratory
Lexington, Massachusetts, USA
{matthias.beebe, maa, paul.foley, denisegalejs, smooney, iulian.popescu, kevin.rottman, meryl.stav}@ll.mit.edu

***Abstract*—As the airborne ISR application space evolves, the quantities of data acquired by remote sensing systems such as radar, electro-optical, and infrared systems are growing larger, and advanced algorithms are imposing more challenging computational requirements for real-time processing. While the difficulties in processing sensor data in real-time is the topic of extensive research, the rapidly shifting technology and application complexity has led to pronounced system lifecycle challenges, including the constant threat of technology obsolescence and unsustainable maintenance costs. One way for Government programs to address this reality economically is to shift the ISR system acquisition strategy to facilitate the timely, cost-effective insertion and upgrade of technologies, through the utilization of an open architecture (OA) approach to system design standards for application ready processors (ARPs). OA design leverages industry-standard hardware and middleware, thus engaging a broader development community and lowering barriers for third-party application development. For this approach to succeed without sacrificing functional capabilities and real-time performance, effective benchmarks are necessary to ensure that an ARP system can meet the mission constraints and performance requirements of real-world applications. This work investigates the measurement of real-time performance of commodity high-speed processing solutions and middleware for airborne systems using OA composite benchmarks, i.e., benchmarks that characterize computational performance of the system as a whole, while also validating OA principles. For ISR systems, processing performance must often be counterbalanced by size, weight and power (SWaP) constraints that often necessitate application-specific configurations (e.g., mapping and scheduling) for system-level optimization. Using ground moving target indicator (GMTI) radar as an example, we demonstrate the use of an open architecture benchmarking framework using industry-standard middleware to indicate the suitability of candidate systems for ISR applications under constrained SWaP.**

Keywords—Open Architecture, ISR; GMTI; Heterogeneous Benchmark; OpenCL; MKL; VSIPL; MPI

I. INTRODUCTION

In airborne high-speed sensor signal processing applications, real-time performance is critical. Both application design and system-level optimization are important to meeting the latency requirements imposed by sensor data rates and usage scenarios. Additionally, for systems with SWaP constraints, fine-tuning application efficiency to maximize hardware utilization is crucial. In the case of airborne radar modes such as ground moving target indicator (GMTI) and synthetic aperture radar (SAR), signal processing algorithms present specific challenges related to data dependency and data distribution for parallelization. All of these needs and constraints often lead to highly optimized, yet tightly integrated (or highly coupled) systems of hardware and software, which comes with significant costs, such as time and effort to maintain, upgrade, or replace, as well as opportunity cost in the form of reduced re-use and adaptability. These costs present a significant issue when it comes to the acquisition and sustainment of systems throughout their lifecycle.

At the same time, recent developments in commodity processor technology for high-performance computing (HPC) such as system-on-a-chip (SoC) processors and graphics processing units (GPUs) for handheld and desktop platforms have led to increased programmability through maturing heterogeneous application programming interfaces (APIs) and languages targeting graphics processors [1]. Examples include the CUDA APIs for development on NVIDIA devices, and the more portable OpenCL APIs which are supported across multiple devices. The prevalence and widespread use of these technologies lend to their accessibility, which is one of the goals of the open architecture paradigm. These developments make it feasible to implement high-performance applications in both homogeneous and heterogeneous environments using multi-threading, multi-tasking, and multi-core computing techniques while moving large amounts of data between distributed nodes. Characterizing the performance of such systems presents a challenge, as kernel-level or network benchmarking techniques typically measure focused capabilities, and may not reflect actual achievable performance across an entire system for a composite application.

In this paper, we present a case study for benchmarking a SWaP-constrained system for GMTI radar applications using an OA benchmarking framework. We discuss the motivations and challenges to characterizing system performance for

Distribution A: Public Release. This work is sponsored by Assistant Secretary of Defense for Research & Engineering under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

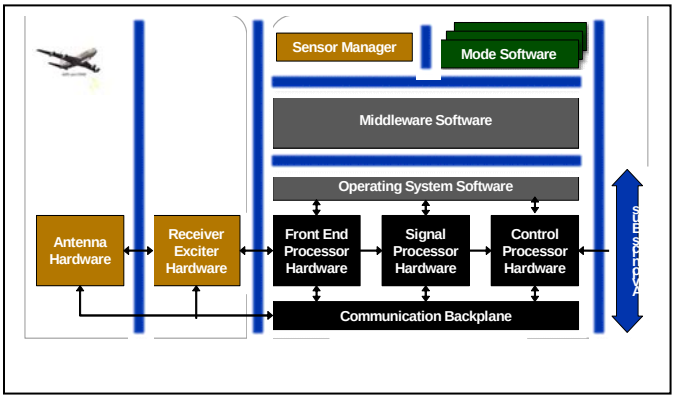
specific HPC applications using an OA approach, introduce the concept of a system-wide or holistic benchmarking application, and provide empirical results from a six-month study involving three commercial vendors. Over the course of the study, each participating vendor proposed and assembled systems adhering to specifications set out in a Government objectives document for ARP solutions. The primary work of the study focused on tuning of the benchmark software for the three solutions proposed and prototyped by the participating vendors.

II. Motivation

A. Open Architecture Goals

Sensing applications in the ISR domain, such as SAR or GMTI require real-time signal processing of large volumes of data within strict SWaP constraints. Historically, this application space has been dominated by expensive, highly specialized and slowly evolving processor technology that is both expensive to build and difficult to maintain. Furthermore, these specialized systems are often prohibitively costly to adapt toward new scenarios or processing algorithms. However, this has started to change in the last decade, and the Government is exploring a different approach to the procurement and maintenance process, designed to ensure that new system components can be easily integrated and upgraded. The principal idea is that by defining key components of an airborne sensor system and clearly defining the interfaces between these components, as illustrated in Figure 1, the architecture of the overall system remains open, and various functional subcomponents can be separately assessed and integrated or replaced. In the case of airborne radar, the result of this approach is to allow third-party radar application development, and to enable low-cost processor and middleware upgrades throughout the radar program lifecycle.

Fig. 1. Open Architecture Radar System



B. Market Trends in High-Performance Computing

With the end of the era of increasing clock-rates in single-core computing [2], parallel processing technologies have become ubiquitous, appearing everywhere from desktop computers to smartphones to supercomputers. Market demand in the commercial computing industry has brought forth a wide array of low-cost, high-volume general-purpose processors and GPUs. As cluster computing and distributed heterogeneous systems becoming the norm in high performance applications, an assortment of middleware technologies for programming parallel platforms has been developed. OpenCL and MPI are examples of industry standard middleware for parallel programming on heterogeneous platforms, and are supported on a wide array processors and communication technologies. Other standards include the vector signal and image processing language (VSIPL) and the data distribution standard (DDS) for inter-process communication, both maintained by the Object Management Group (OMG). Some middleware technologies are not officially standardized by any organizing body, yet remain viable from an open architecture point of view, through their wide adoption and prevalence in the marketplace. Examples of these include the math kernel library (MKL) developed by Intel, Inc., and the CUDA programming language offered by NVIDIA for programming their GPU products. OpenVPX is a switched fabric standard developed specifically for high-performance applications in 3U and 6U configurations often used in SWaP constrained environments, and is supported by a wide array of vendors.

Regardless of level of standardization or adoption in the marketplace, the reality is that modern high-performance systems are multicore, multiprocessor, often heterogeneous processing environments that require multiple middleware and hardware technologies to interact efficiently and predictably. Combined with the expanding demands of airborne ISR sensing applications, system designers need a way to confidently characterize the performance of candidate systems for the applications at hand. Many benchmarks exist for measuring processing, communication, and middleware performance. For example, a wide array of FFT and other math kernel benchmarks are available, similarly for communication fabric benchmarks such as `netperf`, and middleware implementations, e.g., `ccmperf` [3,4]. These tools are useful for differentiating processors and network fabrics via relative benchmark scoring. However, benchmarking system components individually may not provide an adequate level of confidence that as a composite whole, the system will meet or exceed critical real-time requirements of a particular application. One way to address this issue would be to port an existing application to candidate systems and measure absolute or relative performance. This approach may be costly however, as porting and mapping an existing application to new hardware and possibly new middleware may require significant effort.

We propose that to mitigate this cost, a composite benchmark application can be used instead, to tie individual benchmark kernels together in a way that reflects a real-world application, yet provides flexibility in mapping computation and configuring parallelism via open standard middleware for computation and communication. This approach may serve as an effective means to confidently characterize the emergent performance of candidate systems for time critical ISR sensing applications.

III. OA Benchmarking Approach

A. Composite Benchmark Goals

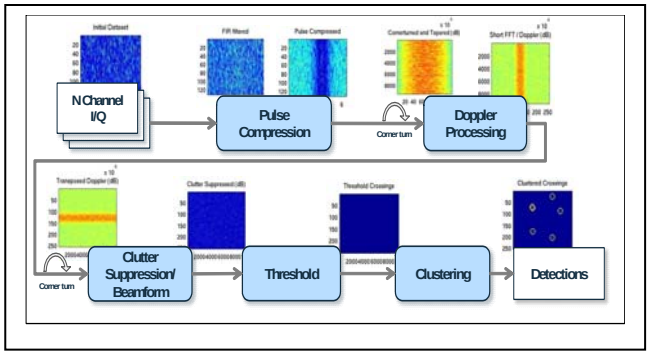
Benchmarking a composite system for a specific application requires flexibility. A benchmark that measures system performance for a composite algorithm chain must be configurable in order to adequately account for communication, data reorientation (transpose operations), distribution and aggregation, and of course mathematical operations for signal processing.

A key distinction of this benchmarking approach is that the sequence of operations defined by the processing chain and output verification software serves primarily as a starting point for the assessment process. The benchmark software must be architected to be easily modified, mapped, and optimized for a particular hardware solution. Open architecture principles within the benchmark itself are critical to the success of this goal in practice.

B. GMTI Benchmark Derivation

In order to characterize overall system performance for specific applications, it is necessary to use actual applications as a basis for the benchmark software used for measurement and assessment. In this case study, we used an algorithm from a fielded pod-class radar application as a basis for the canonical GMTI benchmark processing chain. This algorithm represents all computations following digital sampling of sensor in-phase and quadrature (I/Q) data for each coherent processing interval (CPI). In order to maintain flexibility and to meet use-case goals for the benchmark application, we selected the most computationally intensive operations from the GMTI processing chain and removed operations that were computationally negligible, with the exception of any operation (such as detection thresholding) that is necessary for forming meaningful output for validation. An illustration of the benchmark processing chain is shown in Figure 2.

Fig. 2. GMTI Benchmark Processing Chain



The principal computational components of this processing chain are pulse compression, Doppler processing, adjacent-Doppler bin space-time adaptive processing (clutter suppression and beam-forming), adaptive median filter thresholding, and detection clustering. Each processing component in the chain presents a unique computational challenge, and also features unique data distribution possibilities. For example pulse compression is comprised primarily of a time-domain FIR low-pass filter, followed by a frequency-domain matched filter requiring long FFTs, and can be distributed arbitrarily across pulses (i.e., a single dimension of the three dimensional input data cube), whereas Doppler processing is data dependent in the pulse dimension, but can be distributed in the range samples dimension.

Upon identifying the components and processing chain for characterizing system performance reflective of the GMTI application, we defined computationally stressing application scenarios, based again on capabilities of actual modern fielded systems. We then used these scenarios to define and synthesize five measurement cases for the benchmark to process. Table I shows some examples of the GMTI radar parameters used in defining the various cases.

TABLE I. GMTI Case Study Radar Parameters

	Case 1	**Case 2**	**Case 3**	**Case 4**	**Case 5**
Range Swath (km)	20	20	25	25	50
PRF (Hz)	1300	1200	1200	1300	1300
# Pulses	24	48	72	72	96
# Samples	15847	15847	19811	41269	99070
# Range Samples	3847	3847	8902	18542	71797

Each of the above parameter sets drove corresponding latency thresholds for real-time performance, based on pulse repetition frequency (PRF) and corresponding data collection times. These latency thresholds were used as goals to achieve on the candidate systems proposed by vendors participating in the study. Verification of processing results is built into the benchmark software, using heuristic measurements (e.g., total number of output clusters, percentage of mismatched threshold crossings, etc.) to assure integrity of the processing chain is maintained throughout any modifications necessary for optimization.

In addition to the performance goals, the candidate systems tested during the study were also required to adhere to specific SWaP constraints dictated by the pod-sized radar applications for which their suitability was being tested. For the GMTI application case study, the SWaP requirement imposed the limits shown in table II.

TABLE II. GMTI Case Study SWaP Constraints (POD Class)

Constraint	**Threshold**
Power*	1400 Watts
Size	11” x 14” x 16”
Weight	70 lbs.

[a.] 240V requirement; lower 28V requirement not shown

Other hardware guidelines stipulated include MIL-STD-461E for radiated emissions, susceptibility, and lightning

protection, and environmental considerations such as vibration performance, operating temperature, operating pressure, etc.

C. Workload Analysis

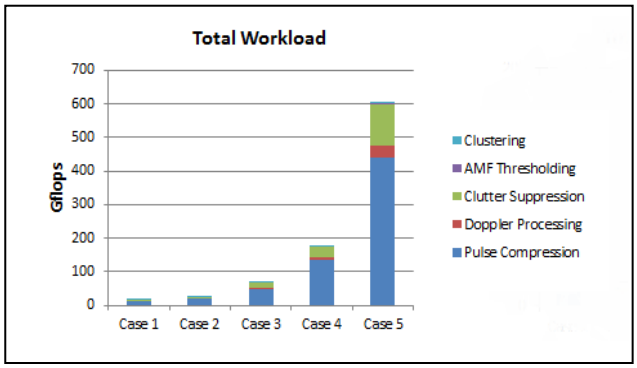
To determine the workload for each benchmark input case, we compiled a detailed floating-point operation (FLOP) count for each computation component at the CPI level, using the same radar parameters that were used for the simulated scans. Workloads for each kernel were assembled based on a combination of well-known asymptotic complexities and nominal FLOP count values for basic operations such as complex multiplications, complex divisions, square roots, etc., using multiple problem sizes.

In general, the following measurements were used to derive and characterize kernel performance on respective processing resources:

- $L_1(k, d_i)$, which is the observed time, or latency, to perform kernel k on a single dataset size d_i using a single processor. This measurement includes both computation time and the time to move the data for the problem from the staging area (off the processor) to a local computation or operation memory.
- $W(k, d_i)$, the workload of a computation on dataset size d_i. This workload is defined in an operation dependent and system-independent way. (For floating-point computation operations, W is the floating-point operation count, while for communication operations, $W(k, d_i)$ is the number of bytes transferred.)

The relative workloads of the computational components in the GMTI processing chain are shown in figure 3.

Fig. 3. GMTI Computation Component Workloads



The pulse compression component accounts for most of the workload in the processing chain, with clustering representing the least workload. However, it should be noted that achievable efficiencies for operations such as pulse compression (primarily FIR filter and long FFTs) are much higher than that of primarily logical operations such as clustering.

Given the above measurements, the derivation of the following performance metrics can be defined:

- $T(k, d_i)$, the sustained processing throughput achieved for a given computation stage k, measured in operations per unit of time.
- $E(k, d_i)$, the efficiency of the computation stage, that is, the use of resources relative to the potential of the processing resource.

The sustained processing throughput for computation stage k is computed as:

$$T(k, d_i) = \lim_{n \to \infty} \frac{nW(k, d_i)}{L_n(k, d_i)}$$

where $L_n(k, d_i)$ is the total time to solve n problems of the given type using the processor. As above, $L_n(k, d_i)$ includes the time to move the data from the system memory into local memory. For more information about sustained processing throughput and some trade-off considerations, see [5].

The efficiency $E(k, d_i)$ of a kernel as attained on a given processor is defined as:

$$E(k, d_i) = \frac{T(k, d_i)}{U(k)}$$

where $U(k)$ is the kernel-dependent theoretical upper bound, or 'peak rated performance', of the processor provided by the manufacturer. When W is in floating-point operations, $U(k)$ is the theoretical peak floating-point computation rate (based on the clock rate and the number of floating-point units). For a communication operation, where workload is defined in bytes, $U(k)$ is the theoretical peak bandwidth between the communicating units.

The above analysis was done for each computation step in the in the GMTI benchmark processing chain in order to compute achieved efficiencies on processing resources allocated to the various components.

D. Middleware Selection

The open architecture GMTI benchmark is designed to take advantage of multiple middleware options for computation in order to provide a means to compare performance across APIs and in some cases multiple implementations, as well as to remain adaptable to future applications. A survey of supported math middleware across vendors participating in the GMTI benchmark study led to the selection of three primary math and signal processing APIs: Open Computing Language (OpenCL), MKL/IPP, and VSIPL. Communication middleware is currently limited to the MPI standard, as this seemed to be the most widely supported API across vendors participating in the study.

These middleware APIs exhibit varying levels of standardization and openness. For example, VSIPL and OpenCL have well-defined standards and support across multiple platforms. VSIPL has many implementations, including TASP VSIPL, Axis VSIPL, and others. Intel's MKL and IPP APIs are not standardized, but this lack of

standardization is counterbalanced by their prevalence in the marketplace and user community.

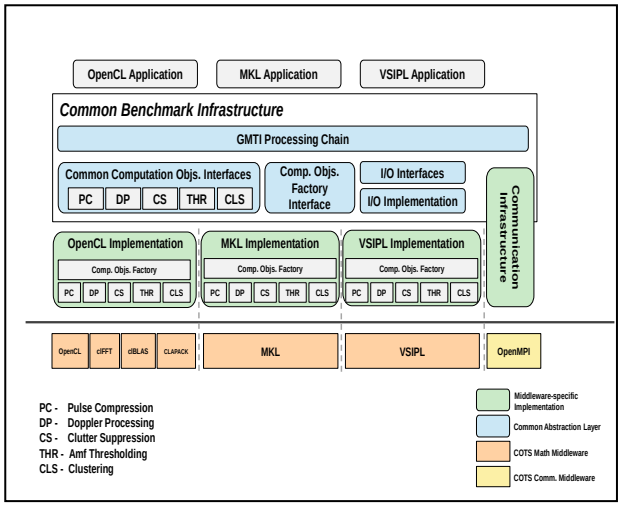
Like the VSIPL standard, MPI has several implementations. MPI implementations that have been tested using the benchmark include Open MPI, MVAPICH, Intel MPI, and others.

E. Software Arhcitecture

The GMTI benchmark software architecture is designed to allow mapping of individual processing components to various hardware configurations via configuration files. Mappings can follow fully data parallel (homogeneous) or task-parallel (heterogeneous) processing schemes.

The benchmark software is also designed to support multiple math and communication middleware implementations via a common object-oriented software hierarchy, with library-specific implementations of the five GMTI benchmark computation kernels being exposed by a *factory interface* design pattern, as shown in the software stack illustration of Figure 4.

Fig. 4. Open Architecture Software Stack



This implementation approach allows users of the benchmark to select which math API is used in performing the application performance measurement. Individual computation kernels built in various APIs can be mixed and matched by users of the benchmark in order to achieve best combinations for performance. For example, in a heterogeneous environment including CPUs and GPUs, a user may choose to run pulse compression and Doppler processing on a GPU using OpenCL or CUDA, and to run the remainder of the processing components on Intel CPUs using MKL. This flexibility in mapping and middleware selection is central to the goal of benchmarking the composite candidate system as a whole.

Fig. 5. Computation Factory Interface (OpenCL variant shown)

```
namespace gmtiBench
{
    OclFactory::OclFactory(
        BenchmarkParameters *parameters,
        TaskConfig *task,
        Const OclDeviceSelectionPolicy *devSelectionPolicy:
        ComputationFactory(),
        m_oclAppEnv(),
        m_devSelectionPolicy(devSelectionPolicy),
        m_parameters(parameters),
        m_taskConfig(task),
        m_pulseCompression(nullptr),
        m_dopplerProcessing(nullptr),
        m_clutterSuppression(nullptr),
        m_amfThresholding(nullptr),
        m_clustering(nullptr)
    {
}
```

Communication within the benchmark software is handled via the MPI standard interface, and the software infrastructure allows for both data parallelism and task parallelism (for heterogeneous processing). The level of parallelism and specific mapping to processing resources is important for the derivation of efficiencies achieved on the candidate systems being tested.

F. Vendor Engagement

The GMTI benchmark study began with the request for information (RFI) submitted to the participating vendors to ascertain availability of candidate systems and middleware solutions for conducting the research. The responses to this request led to the selection of the middleware APIs featured in the benchmark software that was distributed to the vendors. For the duration of the study, we provided technical support for the benchmark software, including various updates and bug-fix patches as well as advice for mapping and optimization to the candidate systems. The vendors in turn provided feedback and insight into the specific challenges of their hardware, such as cache/memory or communication backplane limitations, or difficulties in effectively harnessing processing power of all subsystems in the case of SoC architectures, e.g., due to difficulty in load-balancing, etc. As a result of this process, we found some surprising behaviors that furthered our understanding of some of the performance considerations being examined. For example, distinct performance differences became apparent between various implementations of VSIPL, and similarly, data communication performance between implementations of the MPI standard varied widely.

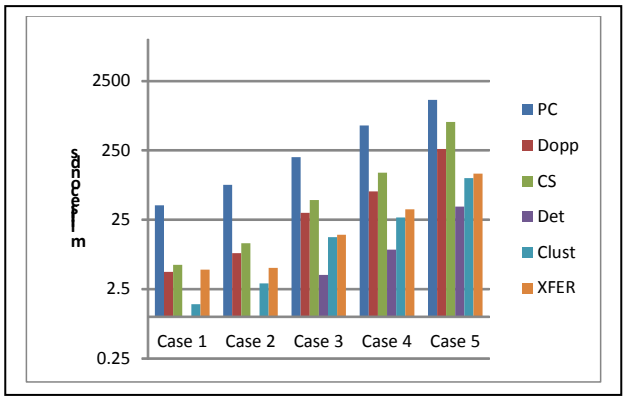
By the conclusion of the GMTI benchmarking study, each of the vendors had developed a fairly comprehensive understanding of the benchmark processing chain and the challenges presented by it at a system level, and had established software mappings most appropriate for their respective systems. Additionally, all parties involved in the study gained appreciation for performance considerations when implementing an application-specific processing chain on embedded system hardware using industry standard APIs.

IV. Example Results

Three unique ARP candidate systems were characterized for performance running the GMTI benchmark under SWaP constraints over the course of the study. While many of the challenges encountered in achieving the goal latency thresholds were common across the proposed solutions, certain distinctions in configuration or underlying processor technology proved to be surprising. For example, small differences in cache size significantly altered performance for certain benchmark data cases. Additionally, polling versus event-driven MPI implementations exhibited distinct behavior when double-buffering was employed between data input processes and data analysis/computation processes.

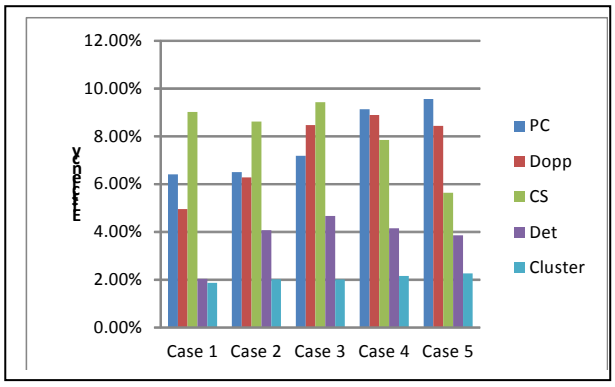
Actual benchmarking study results from the vendors are not shown here to protect competition sensitive information. The following results are an example from lab tests of the GMTI benchmark (MKL) on our laboratory reference system, an 8-node cluster of Intel Xeon X5675 hex core processors clocked at 3.07 GHz, with 12 MB cache and 48 GB RAM per CPU. All nodes are connected via a 40Gb Infiniband backplane.

Fig. 6. Example GMTI Benchmark Component Latencies



Average latencies for each processing component of the processing chain are shown, as well as average overall transfer time. Efficiencies calculated using the workload analysis and equations described in section III.C are also shown in figure 7.

Fig. 7. Example GMTI Benchmark Component Efficiencies



V. Future Work

This work examined several important considerations for evaluating application ready processors for SWaP-constrained remote-sensing applications. Going forward, we are establishing a system integration laboratory for assessing commercial off-the-shelf hardware, middleware, and development tools for developing ISR applications. Toward this end, as a follow-up to the study, several systems are being procured for continued evaluation. While the six-month processor study described here shows that a holistic, open architecture benchmarking approach can be beneficial for characterizing the performance of candidate systems, there are many more variables to explore for candidate system performance assessment. For example, while participants in the study experimented with both hyperthreading and resource oversubscription, results require further analysis. In addition, further configuration is available via MPI build options, MKL threads, vector computation tuning using advanced vector extensions, or operating system kernel tuning. The five benchmark test cases used in this study were designed to fit appropriately within the scope of the work and duration of the effort. Additional benchmark test cases should also be explored, including the design of new synthetic datasets with varying degrees of difficulty and characteristics, such as increased target counts, overlapping targets, different noise models, and jamming scenarios, for example. Having an open architecture benchmarking infrastructure in place to represent real-world applications provides added confidence in assessment, and more rapid evaluation of candidate systems for ISR signal processing applications.

References

[1] K. Olukotun and L. Hammond, “The future of microprocessors,” Queue, vol. 3, no. 7, pp.26-29, Sep. 2005.

[2] H. Sutter, “The free lunch is over: A fundamental return toward concurrentcy in software,” Dr. Dobbs Journal, vol. 30, no. 3, pp. 202-210, 2005.

[3] Benson, Thomas M., “A system-level optimization framework for high-performance networking”, High Performance and Embedded Computing Conference, 2014.

[4] Krishna, Arvind, et al., “CCMPerf: A Benchmarking Tool for CORBA Component Model Implementations”, Real-Time and Embedded Technology and Applications Symposium, 2004.

[5] HPEC Challenge: (Web). http://www.omgwiki.org/hpec/files/hpec-challenge/metrics.html, 12 October 2014.